Relatório Final

Agentes e inteligência artificial distribuída

Daniel Pimenta, ei07020

Diogo Costa, ei09116

Soraia Ferreira, ei09119

Resumo

Este relatório pretende documentar a replicação e parametrização de um sistema multi-agentes para a simulação de um ecossistema, tendo como base o artigo:

C.J.Scogings, K.A.Hawick and H.A.James, Tuning Growth Stability in an Animat Agent Model, Proc. IASTED Int. Conf on Applied Simulation and Modelling (ASM 2007), Aug 29-31 2007, Palma de Mallorca, Spain.

Este modelo simula um ecossistema de animats (animal automata) e estuda de que forma as interacções locais predador-presa, e os comportamentos emergentes associados, contribuem para a estabilidade global do modelo.

Para implementar este modelo foi usada a framework Repast 3. Os resultados desta implementação são comparados com os do artigo final.

Conteúdo

Resumo 2
Introdução 4
Enquadramento 4
Objectivo 4
Estrutura do relatório 4
Enunciado 5
Descrição do cenário 5
Forma de avaliação 6
Especificação 7
Arquitectura dos agentes 7
Estado do mundo e percepções 8
Módulos 9
Protocolos de interacção 10
Desenvolvimento 11
Framework e ferramentas usadas 11
Estrutura da aplicação 12
Diagrama de classes 14
Experiências 16
Estabilidade da população 16
Emergência de tribos 16
Uso das regras 17
Padrões espaciais de agentes 18
Conclusões 18
Melhoramentos 19
Bibliografia 20
Apêndice A – Manual do utilizador 20

Introdução

Enquadramento

O desenvolvimento desta aplicação surge no âmbito da disciplina de Agentes e Inteligência Artificial Distribuída, integrada no plano de estudos do Mestrado Integrado em Engenharia Informática e Computação da Faculdade de Engenharia da Universidade do Porto.

O seu desenvolvimento está associado à programação de agentes distribuídos, utilizando a Framework REPAST, que assenta na linguagem de programação JAVA para a simulação de sistemas multi-agentes.

Objectivo

O principal objectivo deste trabalho consiste em simular um ecossistema através da implementação de um Sistema Multi-Agente, com particular interesse nas estruturas espaciais dos agentes assim como a estabilidade global das populações.

É nosso trabalho descrever correctamente o modelo e posteriormente implementar o mesmo. Desta implementação pretendemos obter resultados semelhantes ao artigo [1].

Estrutura do relatório

Este trabalho encontra-se estruturado em 6 capítulos dos quais, o primeiro é composto por esta introdução.

No segundo capítulo é descrito o problema a resolver, neste caso o trabalho subjacente ao artigo referido anteriormente, a forma de avaliação do trabalho.

No terceiro capítulo são descritas as ferramentas usadas na implementação do sistema multi-agente assim como a implementação propriamente dita.

No quarto e quinto capítulo são documentadas algumas experiências com o SMA implementado assim como conclusões e observações às mesmas.

O último capítulo contém as conclusões gerais deste trabalho e apresenta algumas perspectivas de desenvolvimentos futuros.

Enunciado

Descrição do cenário

O presente trabalho baseia-se no artigo escrito por Scogings, James e Hawick [1]. Este artigo relaciona a forma como interacções simples entre agentes, e os fenómenos emergentes associados, contribuem para a estabilidade global de um modelo específico. Este modelo específico trata-se de um ecossistema de animats com relações do tipo predador-presa.

No modelo desenvolvido, os animais comem, caçam, geram novos agentes e deslocam-se em qualquer direcção num mundo plano e sem fronteiras. Os tipos de agente presentes neste modelo são as "raposas" (predadores) e os "coelhos" (presas). Na base da sobrevivência das raposas está uma alimentação que consiste em comer coelhos e a sobrevivência dos coelhos é assegurada por alimento permanentemente acessível ("erva") para se alimentarem.

O comportamento de cada animal, e consequentemente as suas acções, é regido por um conjunto de regras com condições e ordem de prioridade. Se as condições da regra são satisfeitas então a regra é executada e mais nenhuma é executada nesse momento no tempo para esse agente. Se as condições da primeira regra não se verificam, é executada a segunda; e assim sucessivamente.

Embora em modelos anteriores as prioridades das regras fossem estáticas, no modelo actual a classe de agentes coelho pode ter prioridades diferentes para as primeiras 5 regras. Estas prioridades são definidas pelos genes do agente coelho. Quando vários coelhos partilham o mesmo gene, ou seja a mesma combinação de prioridades de regras, dizemos então que esses agentes coelho pertencem à mesma tribo.

Estas regras usam o factor energia/fome dos agentes e a localização dos seus vizinhos para condicionar as acções dos mesmos. Cada agente possui um inteiro que simboliza a energia do animal, este valor é definido com base num valor máximo parametrizável. Se o animal não come, a energia do animal é decrementada e quando este come ela retorna ao valor máximo. Considera-se que um animal tem fome se o valor da energia for inferior a 1/3 da energia máxima definida e se a energia chegar a zero, o animal morre de fome.

A reprodução de agentes não usa nenhuma combinação genética ou mutação. Em vez disso, para cada novo agente, os seus genes são uma réplica exacta dos genes de um dos progenitores, sendo que esta escolha é feita de maneira aleatória. No caso das raposas visto não existirem diferenças de genes, o resultado são descendentes que consistem em cópias exactas de ambos os progenitores, no caso dos coelhos isto já não é verdade.

É importante referir que o modelo não obriga ao uso exclusivo de uma célula por apenas um agente. De facto a procriação coloca um novo agente na mesma célula que um dos seus progenitores [2,3]. De referir também que o espaço é expandido automaticamente para evitar a interferência dos limites do espaço na simulação.

O modelo tem também o conceito de idades, cada agente começa com uma idade de 0 e por cada iteração da simulação, envelhece um ano, eventualmente este morrerá quando a sua

idade for maior ou igual a uma idade parametrizável. Os coelhos têm uma idade máxima de 20 timesteps e as raposas de 50 timesteps. [3]

De acordo com o artigo [3], a procriação de um animat é também afectada pela quantidade de animats à sua volta. O “crowding limitation” limita a reprodução a um determinado agente, se à sua volta existirem outros n agentes do mesmo tipo. No caso das raposas este valor é de 3 e no caso dos coelhos é de 5 [3].

Forma de avaliação

Em primeiro lugar, e como o nome do artigo original implica, teremos de ter populações de agentes, que apesar de poderem sofrer flutuações no tempo, se mantenham em grande parte estáveis.

Esperamos também que poucas tribos consigam sobreviver no longo prazo e que essas tribos mais bem sucedidas partilhem os mesmos genes das tribos mais bem sucedidas no artigo.

É também esperado que certas regras sejam cumpridas com mais frequência que outras. Mais uma vez usaremos o artigo [1] como referência.

Especificação

Arquitectura dos agentes

Os agentes usados neste trabalho seguem uma arquitectura reflexiva. Cada agente tem um conjunto de percepções obtido a partir do estado do mundo num dado instante de tempo, estas percepções são então mapeadas para acções por regras.

Com esta abordagem é esperado que a partir de regras simples, ao nível microscópico, replicadas por diversos agentes, que surjam novos comportamentos no plano macroscópico, sendo que nem sempre estas relações são evidentes.

Neste contexto existe uma hierarquia associada às regras, na medida em que apenas uma pode ser executada em cada instante de tempo. Isto significa que para várias regras despoletadas simultaneamente por percepções, apenas a de maior prioridade ou hierarquia vai executar e causar uma determinada acção. Na tabela 1 temos a lista de regras para cada tipo de agente no modelo.

Tabela 1

Coelho (presa)	Raposa (predador)
• **Foge de raposa adjacente** • **Foge de coelho adjacente se tem fome** • **Procria se não tem fome e existe coelho adjacente** • **Aproxima-se de coelho se não tem fome** • **Probabilidade de 50% em mover-se aleatoriamente** • **NOP**	• Come coelho se tem fome e coelho adjacente • Aproxima-se de coelho se tem fome • Procria se não tem fome e existe raposa adjacente • Aproxima-se de raposa se não tem fome • Probabilidade de 50% em mover-se aleatoriamente • NOP

Apesar de os agentes raposa terem as mesmas regras com as mesmas prioridades, no caso dos coelhos, todas as prioridades (com excepção da regra NOP) são definidas pelos genes individuais de cada agente coelho. A colecção de genes possíveis para os agentes coelho consiste numa permutação específica das 5 primeiras regras.

Então, sabendo que queremos permutar as 5 primeiras regras (120 combinações possíveis), e sabendo o número inicial de coelhos é N, então começaremos aproximadamente com $[N/120]$ coelhos para cada gene colocados aleatoriamente no espaço da simulação. Este valor N é definido pelo utilizador como parâmetro da simulação.

Na Figura 1 agrupamos as várias tarefas em 3 grupos, cada agente deverá saber representar-se no espaço, através do módulo GUI, deverá também de interpretar o mundo através dos seus sensores e alterar o mesmo através de actuadores. Por fim o agente necessita de determinar que actuadores usar em que casos e de que maneira, isto é trabalho para os vários módulos existentes no agente e variam para as presas e predadores.

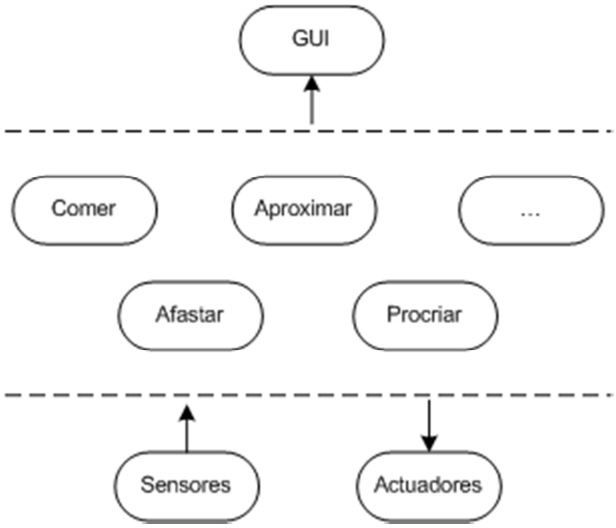


Ilustração 1

Estado do mundo e percepções

Energia

Todos os agentes sem excepção competem por energia, esta energia está disponível para as raposas sob a forma de coelhos, e para os coelhos sob a forma de erva sempre disponível pelo mundo. Todos os agentes tem uma energia máxima definida pelo utilizador no inicio da simulação, e serve de limite para a energia. Esta irá ser decrementada sempre que o agente não coma. Quando o agente come, a energia é restaurada ao máximo. Se por acaso a energia atingir o valor zero, o agente morre.

Coelho/raposa adjacente

Consideramos que um agente B está adjacente a um agente A se este se encontra na mesma célula ou numa das oito células directamente adjacentes num autómato celular de duas dimensões como é o caso na Figura 2.

Isto equivale portanto a acharmos os "Moore Neighbours" desse agente A.

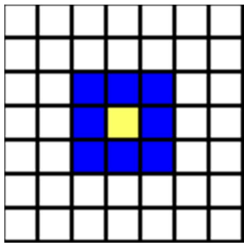


Ilustração 2

Coelho/raposa mais próximo

Para encontrarmos o agente B mais próximo de A usamos a "Extended Moore Neighborhood", que é o caso geral da "Moore Neighborhood", pois é usado um raio para determinar a vizinhança (Figura 3).

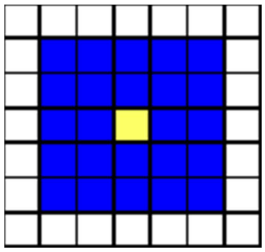


Ilustração 3

Definimos então um método no agente que recebe um inteiro e devolve um agente. Caso o inteiro seja maior que o raio máximo é retornado o valor *null* como resultado, o que significa que não foram encontrados agentes próximos. Caso o raio seja inferior ao máximo então procuramos na vizinhança com esse raio por agentes, caso seja encontrado pelo menos um, esse é imediatamente devolvido, caso nenhum seja encontrado a função é chamada recursivamente para o raio + 1.

De notar que tomamos esta abordagem, pois os agentes não têm obstáculos e podem-se movimentar livremente em qualquer direcção. Caso existissem obstáculos esta solução não seria adequada.

Módulos

Comer coelho

Este módulo está presente nos agentes Raposa e é despoletado pelas percepções:

- Agente tem fome;
- Coelho adjacente.

Este módulo detecta se existe algum coelho adjacente, caso não exista o módulo falha e não é despoletada nenhuma acção. Caso seja encontrado um coelho adjacente, a raposa assinala o coelho como morto e sinaliza a sua energia com a energia máxima.

Aproximar/Afastar de coelho/raposa

Estes módulos estão presentes nos agentes Raposa e Coelho e dependendo do agente são activados por percepções diferentes.

Genericamente falando o agente A descobre primeiro o agente B mais próximo, dependendo do caso, o agente A segue o vector definido pela posA - posB ou pelo vector definido pela posB - posA, afastando-se ou aproximando-se de B respectivamente.

Apenas como nota este processo é um pouco mais complexo, pois temos de ter em conta que o espaço, sendo um torus, nem sempre o caminho mais rápido de A para B é no sentido de A para B, mas em alguns casos no sentido contrário.

Procriar

Este módulo está presente nos agentes Raposa e Coelho e é despoletado pelas percepções:

- Agente não tem fome;
- Agente do mesmo tipo está adjacente.

Este módulo extrai a sequência genética de um dos agentes envolvidos na procriação, cada um destes agentes tem 50% de hipóteses de ser escolhido.

Quando um novo agente é criado, apenas tem 1/3 da energia máxima, isto significa que todos os agentes tem de começar por procurar comida. Isto ajuda a evitar a perpetuação de genes devido exclusivamente à procriação.

Mover aleatoriamente em 50% dos casos

Caso este módulo seja activado existe 50% de hipóteses de o agente se mover aleatoriamente. Esta regra garante que existe alguma exploração aleatória do espaço por parte dos agentes.

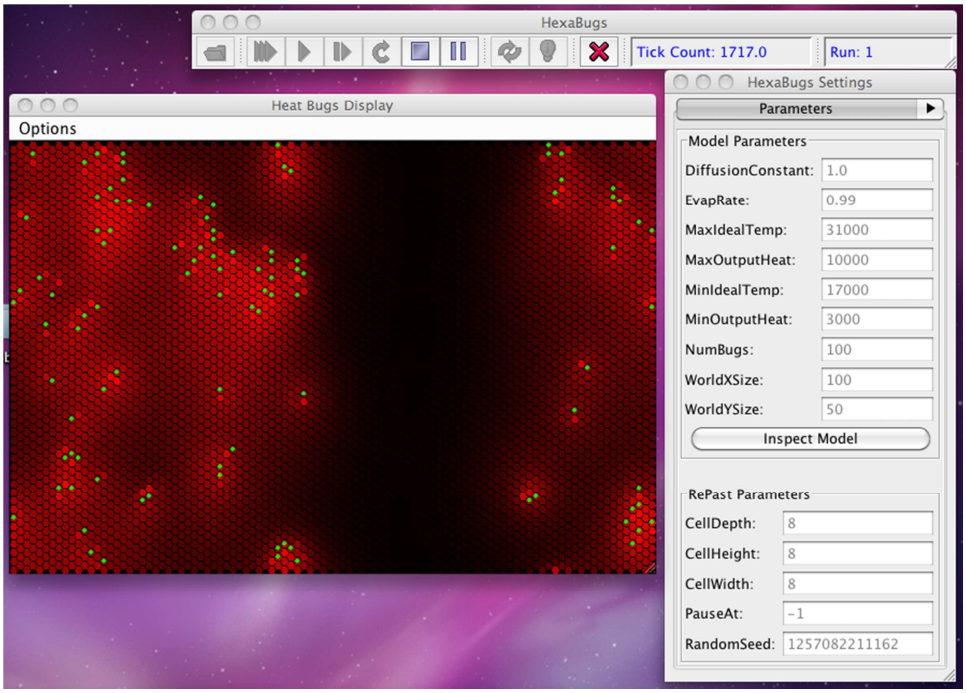
Protocolos de interacção

Neste caso o SMA não depende da troca de mensagens entre agentes, mas sim de decisões baseadas no local para cada agente, como por exemplo o nível de energia, ou o número de vizinhos etc. Como tal protocolos de interacção e trocas de mensagens não foram necessárias neste trabalho.

Desenvolvimento

Framework e ferramentas usadas

O REPAST é uma ferramenta de código aberto que visa apoiar o desenvolvimento de modelos extremamente flexíveis de agentes em "sociedade". Esta é talvez uma das ferramentas mais utilizadas na simulação multi-agente. Existem outras ferramentas de simulação também muito utilizadas como é o caso do NetLogo, Swarm, AnyLogic, Mason e Ascape que não serão usadas neste trabalho.



REPAST 3 tem uma variedade de recursos, onde se destacam as seguintes:

- Diversos modelos de agentes de exemplo;
- Totalmente orientado a objectos;
- Contém ferramentas gráficas e de log para os resultados de simulação;
- Framework Simulador Monte Carlo;
- Visualizador bidimensional dos agentes;
- Permite o acesso e modificação das propriedades dos agentes, os comportamentos e propriedades do modelo, em tempo real;
- Inclui bibliotecas de algoritmos genéticos, redes neuronais, geração de números aleatórios e matemática especializada;
- Modelos integrados de modelagem dinâmica;
- Ferramentas de modelagem de redes sociais;
- Sistema Geográfico de Informação (GIS) ;
- É totalmente implementado numa variedade de linguagens incluindo Java e C\#;
- Os modelos podem ser desenvolvidos em várias linguas como Java, .NET, Managed C++, Lisp, Prolog e Python;
- Disponível em praticamente todas as plataformas, incluindo Windows, Mac OSX e Linux. O suporte inclui tanto os computadores pessoais e em grande escala de clusters de computação científica.

A Framework REPAST, permite-nos desenvolver simulações de SMA, de uma maneira fácil, pois grande partes das funcionalidades necessárias para tal já fazem parte do REPAST, como a geração visual dos agentes, recolha de dados sobre o SMA, visualização em gráficos desses dados, controlo temporal da simulação, captura de vídeo e imagens, e definição de parâmetros do SMA.

O IDE usado foi o Netbeans, ambiente de desenvolvimento integrado multi-plataforma que permite criar aplicações escritas em diversas linguagens, entre elas JAVA e visa simplificar o desenvolvimento e aumentar a produtividade de quem desenvolve as aplicações.

Estrutura da aplicação

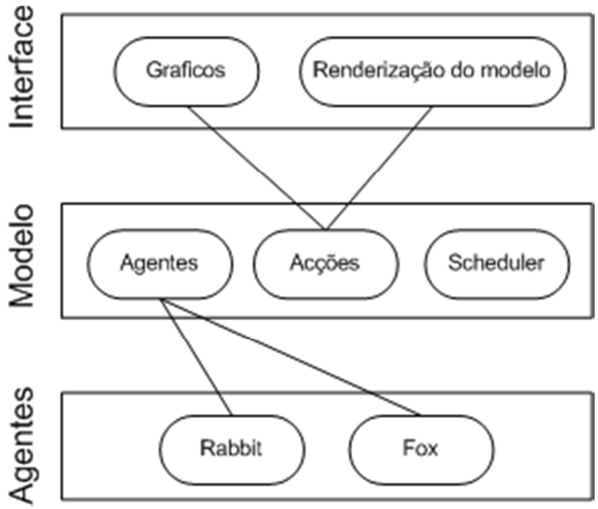


Ilustração 4

O sistema multi-agente desenvolvido está estruturado em 3 grandes níveis:

- Interface
- Modelo
- Agentes

A interface é totalmente disponibilizada pela Framework Repast e é opcional. É responsável por representar visualmente a informação dos níveis inferiores, modelo e agentes, sob a forma de gráficos ou de uma representação visual da posição/informação dos agentes.

O modelo é desenvolvido em Java utilizando classes ou implementando interfaces do Repast e tem informação sobre o estado do mundo, agentes, scheduler e acções. O modelo é responsável por criar os agentes correctamente, criar um scheduler e definir acções a serem despoletadas pelo scheduler, permitir a parametrização do modelo com base em inputs do utilizador, e criar um espaço para os agentes assim como uma visualização associada.

Os agentes implementados são definidos pelas classes Fox e Rabbit para presa e predador respectivamente. Estas classes irão encapsular os métodos para a criação das percepções, módulos e acções de cada tipo de agente.

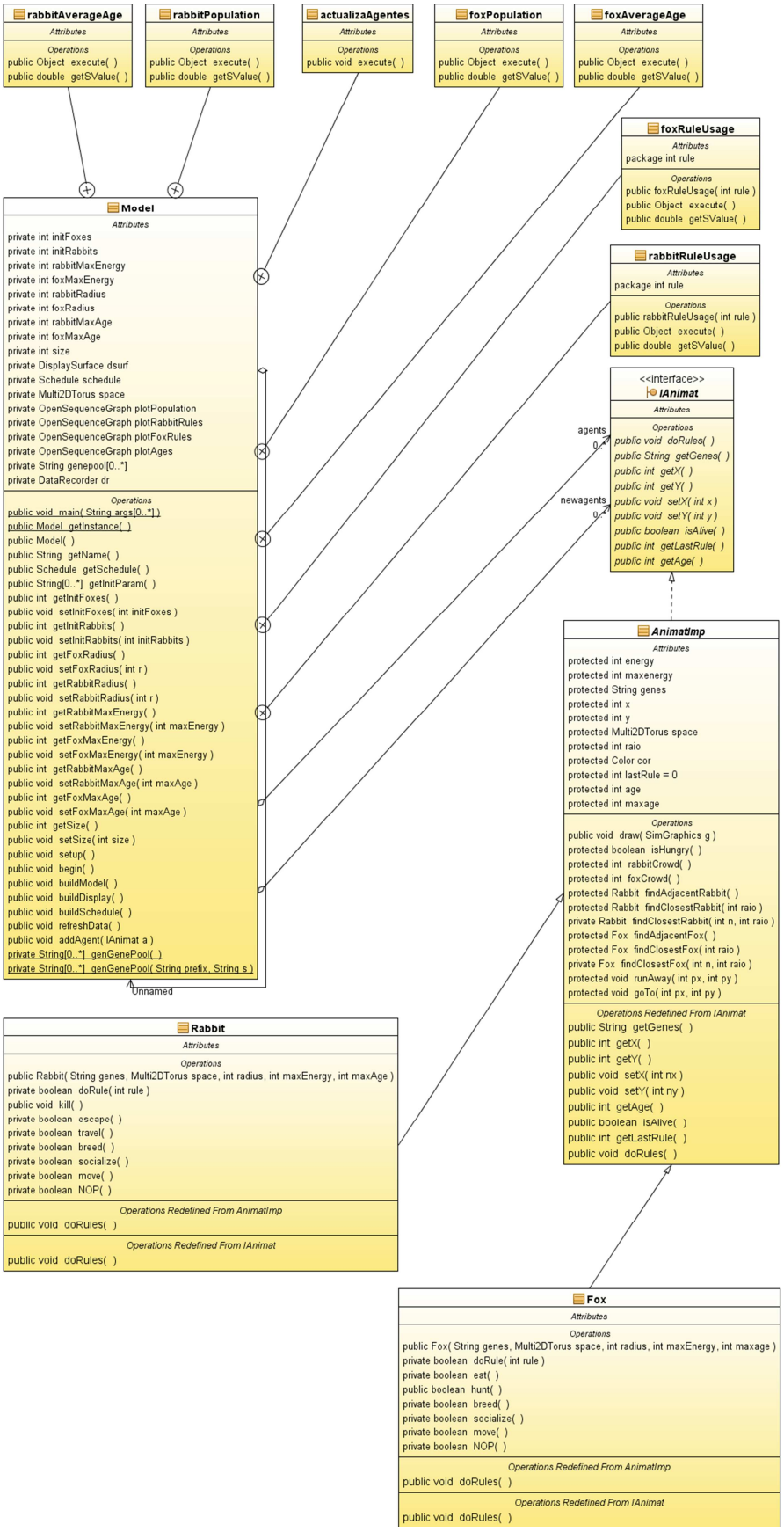
Uma vez que o simulador é discreto, o modelo evolui no tempo também de forma discreta através do scheduler. Este tem acções associadas e em cada unidade de tempo despoleta essas acções.

É através de uma destas acções que todos os agentes aplicam as suas regras num dado instante no tempo, de notar que garantimos que a ordem pela qual os agentes aplicam as suas regras nesse instante é totalmente aleatória e é também através de uma acção que refrescamos a visualização dos agentes ou os gráficos correspondentes na interface.

De uma maneira simplista a aplicação é semelhante ao seguinte pseudocódigo [4]:

```
Require: Choose random number seed
  Initialise Animat Configuration
  Set up spatial fox pattern: uniform random genes
  Set up spatial rabbit pattern: single standard gene
  for each time step do
    if total animat extinction then
      break
    end if
    List current live animats
    for each animat in a random order do
      update animat according to its rules
    end for
    Add new births to live list
    Garbage collect dead animats
    Record time step statistics
  end for
```

Diagrama de classes



Classe Model

Esta classe é a classe principal do projecto. A mesma contém o método main() onde são inicializados os objectos Fox e Rabbit. Esta herda da classe SimModelImpl do Repast3 e inicializa-o para simulação.

Classe ActualizaAgentes

Esta classe tem como função actualizar os objectos Fox e Rabbit no modelo. Tem um método execute, que é executado a cada ciclo de simulação.

Classe FoxRuleUsage e RabbitRuleUsage

Classes que actualizam os valores para a representação dos gráficos sobre a percentagem das regras aplicadas para cada objecto Fox e Rabbit.

Classe FoxAverageAge e RabbitAverageAge

Classes que actualizam os valores para a representação dos gráficos sobre a média de idades dos objectos Fox e Rabbit.

Classes FoxPopulation e RabbitPopulation

Classes que actualizam os valores para a representação dos gráficos sobre o número de objectos Fox e Rabbit, população.

Interface IAnimat

Esta interface define os métodos que todos os animats no modelo tem em comum, como por exemplo getters e setters para as coordenadas, genes, idade, verificar se o animat está vivo etc.

Classe AnimatImpl

Implementação da interface IAnimat. Esta classe implementa os métodos e atributos que são comuns a todos os animats. Energia, idade, genes são apenas alguns dos atributos. A classe também define métodos para consulta dos vizinhos.

Classe Fox

Este objecto contém o corpo dos métodos para a aplicação das regras, genes, das raposas. Esta herda do objecto AnimatImpl.

Classe Rabbit

Este objecto contém o corpo dos métodos para a aplicação das regras, genes, dos coelhos. Esta herda do objecto AnimatImpl.

Experiências

Estabilidade da população

População aos 500 e 1000 timesteps, para um espaço de 300x300 e 100x100 respectivamente.

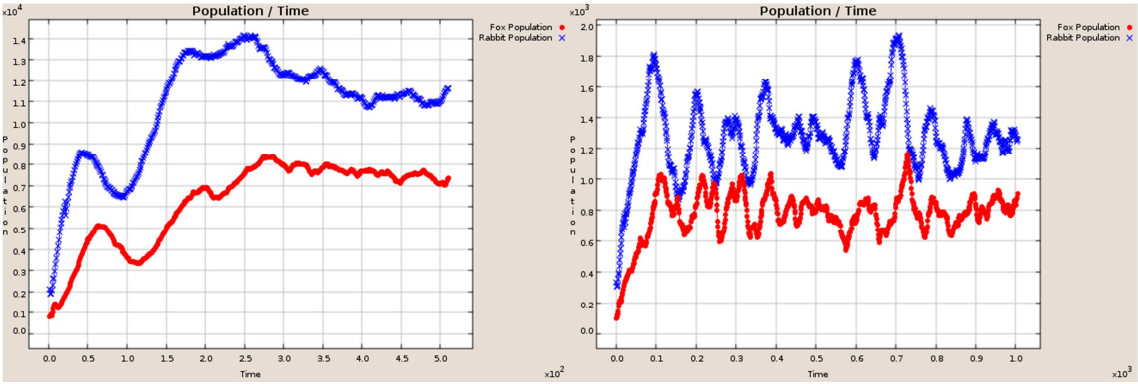


Ilustração 5

Emergência de tribos

Tribos mais prevalentes aos 1000 timesteps, para um espaço de 100x100.

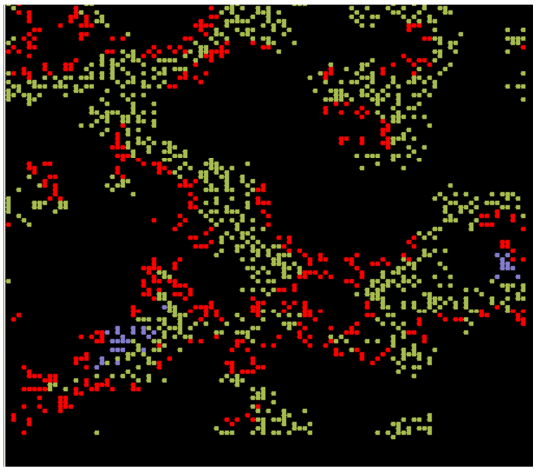


Ilustração 6

Após 500 timesteps num espaço de 300x300 a quantidade de tribos é consideravelmente mais reduzida relativamente ao número inicial. De notar que grande parte do ecrã é ocupada pela tribo esverdeada como anteriormente.

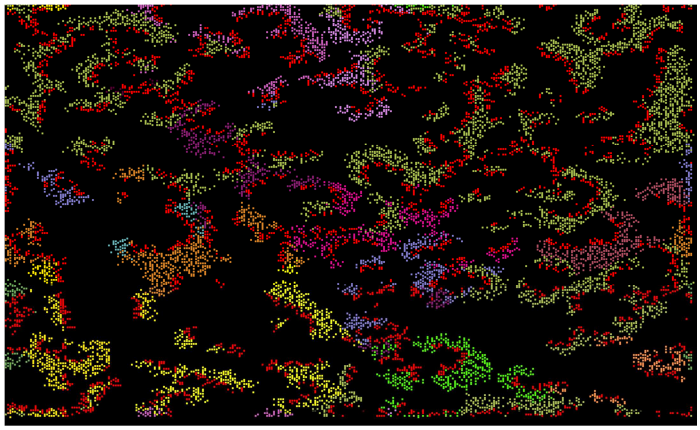


Ilustração 7

Uso das regras

Utilização de regras ao fim de 500 timesteps para um espaço 300x300

Tabela 2

Fox Rule Usage %		Rabbit Rule Usage %	
Eat Rabbit	7	Move away from fox	9
Move towards rabbit	43	Move away from rabbit	60
Breed	10	breed	12
Move towards fox	36	Move towards rabbit	8
Move randomly	4	Move randomly	8
NOP	0	NOP	3

Utilização de regras ao fim de 1000 timesteps para um espaço de 100x100

Tabela 3

Fox Rule Usage %		Rabbit Rule Usage %	
Eat Rabbit	8	Move away from fox	9
Move towards rabbit	46	Move away from rabbit	58
Breed	13	breed	15
Move towards fox	31	Move towards rabbit	9
Move randomly	2	Move randomly	6
NOP	0	NOP	3

Padrões espaciais de agentes

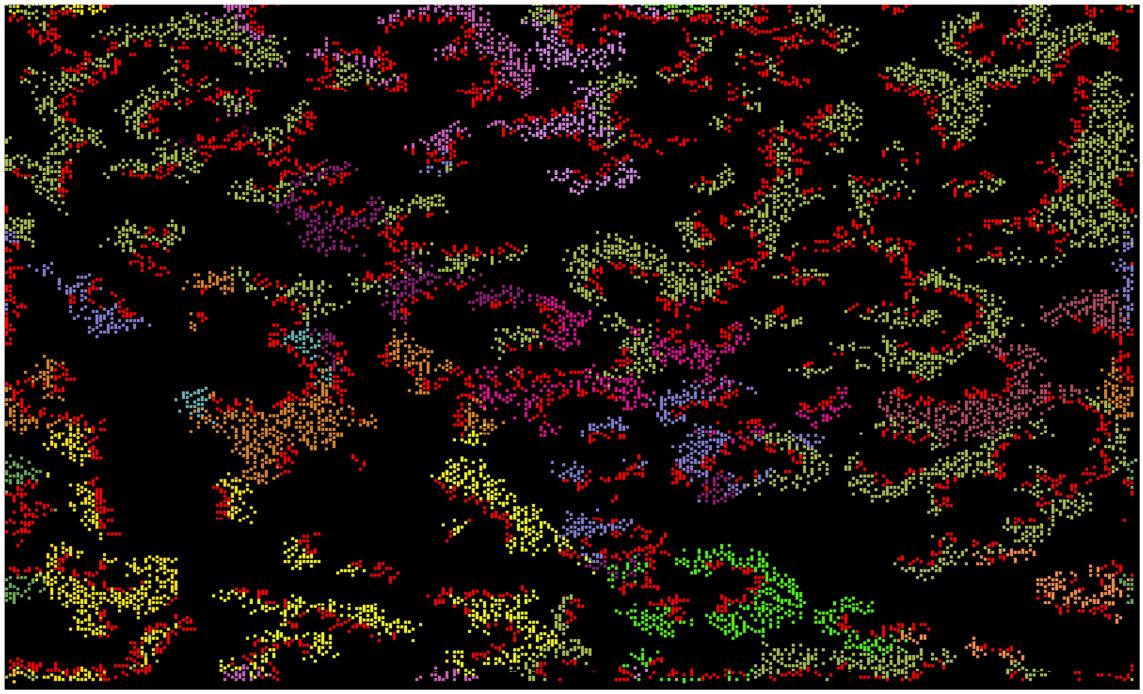


Ilustração 8

Conseguimos observar alguns dos padrões elaborados em [2], como por exemplo as “waves”, “blobs” e “spirals” ou “curves”.

Emergência das tribos

Dos testes que corremos, parece existir uma forte tendência para alguns genes sobreviverem mais frequentemente do que outros. Comparando a imagem 6 e 7 podemos ver que em dois testes totalmente diferentes a tribo mais predominante é a esverdeada. No entanto em espaços muito pequenos isto depende muito da posição inicial dos agentes, e as populações variam demasiado para haver um conjunto de tribos que apareça sempre como dominante.

No entanto para espaços e populações grandes O modelo torna-se menos sensível a estas flutuações e o mesmo conjunto de tribos aparece com mais frequência.

Neste caso começamos com 120 tribos, cada uma com 19 coelhos e, temos cerca de 10 tribos apenas ao final do timestep 500.

Conclusões

Estabilidade da população

Conseguimos em espaços de vários tamanhos manter uma população estável. A estabilidade na nossa opinião, advêm de dois aspectos cruciais no modelo, i)os coelhos so podem procriar se não tiverem fome, e isto só acontece quando não estão com excesso de população, ii)As raposas apenas comem coelhos quando tem fome.

De notar que parece existir uma relação entre a população de presas e predadores. Quando existe um acréscimo ou decréscimo de presa, existe também, com algum atraso, um acréscimo ou decrésimo à população dos predadores. Isto vai ao encontro do artigo original [1].

A estabilidade da população é muito afectada também pela energia máxima dos animais, Se algum deles tiver uma energia máxima muito maior que o outro, esse primeiro consegue reproduzir com sucesso muito mais vezes, pois raramente está com fome, ou leva muito tempo a ficar com fome.

De notar também que populações menores, são mais susceptíveis de flutuações no numero de indivíduos. Isto é visível na imagem 5.

Genes

Na imagem 8 os genes das tribos mais bem sucedidas são:

12354 – Esverdeado

32541 - Verde

31254 – Amarelo

32451 – Castanho

12345 – Roxo

As tribos mais bem sucedidas tem a procriação, a auto preservação e a procura de comida com prioridades muito grandes, aparecendo o movimento aleatório e a procura por companheiro na maior parte das vezes em últimos lugares.

Parece ir ao encontro do que foi ilustrado no artigo original, mas não tivemos tempo para mais testes do modelo para tirar conclusões relativamente a isto.

Uso das regras

Na nossa implementação tivemos valores de uso das regras bastante diferentes do artigo original.

Ao invés de termos as regras NOP e Random move como as mais usadas, tivemos a “move away from rabbit” e “breed” como as mais frequentes nos coelhos.

Parece-nos que estes valores fazem sentido porque as regras “NOP” e “Random Move” se aparecerem no fundo das prioridades, e só vão ser executadas se todas as condições falharem das regras anteriores falharem.

Melhoramentos

As condições de procriação parece-nos ainda não ser as melhores, parece-nos ter de existir um rácio de nascimento que interfira na probabilidade de procriação dos animats, pois temos a procriação a acontecer com mais frequência do que no paper [1]. Se conseguirmos baixar o ritmo de nascimento, e aumentarmos a energia disponível para os animais esperamos que estes comecem a aplicar mais as regras 5 “random move” e 6 “nop” á semelhança do artigo.

Parece-nos também uma boa ideia substituir os factores de sobrepovoamento por relva que quando comida demore um certo tempo a crescer, como naturalmente acotneceria. Temos esperanças que isto tenha o mesmo efeito que o factor de sobrepovoamento na procriação, pois levaria os coelhos a procurar com mais frequência comida.

Bibliografia

[1] C.J.Scogings, K.A.Hawick and H.A.James, Tuning Growth Stability in an Animat Agent Model, Proc. IASTED Int. Conf on Applied Simulation and Modelling (ASM 2007), Aug 29-31 2007, Palma de Mallorca, Spain.

[2] Hawick, K.A., James, H.A. and Scogings, C.J., A Zoology of Emergent Life Patterns in a Predator-Prey Simulation Model Proc. IASTED International Conference on Modelling, Simulation and Optimization (MSO 2006), Gabarone, Botswana.

[3] Hawick, K.A., Scogings, C.J. and James, H.A., Defensive Spiral Emergence in a Predator-Prey Model, In Proceedings of Complexity 2004, Cairns, Australia, December 2004, Edited by Russel Stonier, Quinglong Han and Wei Li.

[4] Hawick, K.A., Scogings, C.J. and James, H.A., Spatial Emergence of Genotypical Tribes in an Animat Simulation Model Computational Science Technical Note CSTN-035 (www.massey.ac.nz/-˜kahawick/cstn/035/cstn-035.html), February 2007.

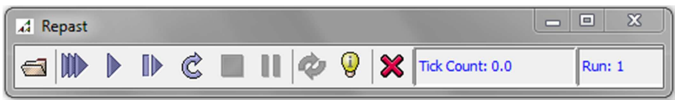
[5] Tools and Techniques for Optimisation of Microscopic Artificial Life Simulation Models, C.J.Scogings, K.A.Hawick and H.A.James, Massey University Technical Note CSTN-033, April 2006.

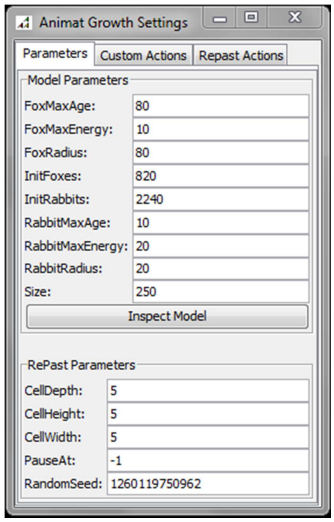
Apêndice A – Manual do utilizador

Para executar o SMA é necessário ter o java runtime instalado e executar o seguinte comando:

```
java -jar AnimatGrowth.jar
```

Ao qual aparecerá o interface do repast. O utilizador poderá então alterar as dimensões do espaço, a energia máxima para as raposas e coelhos assim como as idades, populações iniciais e raio de visão.





Para correr a simulação, basta carregar no botão start ( ).